Logistic Regression

Introduction

Oh!, a wine factory is going to receive a new pack of different wines and they do not have their type labelled (red or white). Ok, don't worry, we can go through each of the wines, look at it color, and label it. But... we would like to do this process automatically. In the following lines, we will face a classification problem to predict if the wine is red or white, depending on its physicochemical attributes.

A classification problem relates input variables x to the output variable y, but now y can take only discrete values, instead of continuous variables as in regression. When y can only take two discrete, it is called binary classification. We will denote these values as $y \in \{0, 1\}$ in the rest of the report, where $0 \equiv$ white class and $1 \equiv$ red type.

Loading the data

The data science pipeline often* needs to split the original dataset into two smaller pieces: the train and test datasets. If we only evaluate our models in the same dataset, the results will be overestimated (aka overfitting). To provide honest assesments of the performance of the predictive models, we will need to validate the models using a test dataset, a partition that has not been used to build the models in order to avoid bias.

(*) Some statistical learning models are robust enough to do not need this division. They can infer the behaviour of the whole population from a sample if some statistical hypothesis are fulfilled.

```
# Loading the dataset into a dataframe
df <- read_delim("../data/processed/wines.csv", ";", escape_double = FALSE, trim_ws = TRUE)

# Train and test dataset, split 80%.
split = nrow(df)*0.8
train = df[1:split,]
test = df[split:nrow(df),]
```

In this case, the test dataset consists of the 20% of the dataset (1300 observations).

Logistic Regression Model

The equivalent linear regression model in classification is the logistic regression model. This model needs to specify a function such that $p(y = 0|\tilde{\boldsymbol{X}})$ and $p(y = 1|\tilde{\boldsymbol{X}})$ are both greater than 0 and sum 1. The logistic function has such properties, defininig the following model:

$$p(y|\tilde{\boldsymbol{X}}, \boldsymbol{\beta}) = \frac{e^{\boldsymbol{\beta}\tilde{\boldsymbol{X}}}}{1 + e^{\boldsymbol{\beta}\tilde{\boldsymbol{X}}}}$$

If $\beta_i > 0$ then increasing one unit in x_i will increase the probability of a success. If $\beta_i < 0$, then the probabilty of success decrease when increasing x_i. When $\beta_i = 0$, $e^0 = 1$, so the odds do not change with x_i.

Full model

We start by defining a logistic regression model with all the 11 attributes as the predictors. We do not use the `quality`, used in regression, and neither the `type`, used as the target variable:

```r
# Logistic regression model with all the variables.
log.full=glm(type~fixed_acidity+volatile_acidity+citic_acid+residual_sugar+chlorides
             +free_sulfur_dioxide+total_sulfur_dioxide+density
             +pH+sulphates+alcohol, data=train, family=binomial)
summary(log.full)
```

```
##
## Call:
## glm(formula = type ~ fixed_acidity + volatile_acidity + citic_acid +
##     residual_sugar + chlorides + free_sulfur_dioxide + total_sulfur_dioxide +
##     density + pH + sulphates + alcohol, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.4910  -0.0567  -0.0230  -0.0002   5.5097
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -2.427e+03  2.284e+02 -10.628  < 2e-16 ***
## fixed_acidity        -9.649e-01  2.529e-01  -3.815 0.000136 ***
## volatile_acidity      5.047e+00  1.110e+00   4.545 5.50e-06 ***
## citic_acid           -2.111e+00  1.364e+00  -1.548 0.121606
## residual_sugar       -9.974e-01  1.105e-01  -9.022  < 2e-16 ***
## chlorides             2.035e+01  4.152e+00   4.901 9.55e-07 ***
## free_sulfur_dioxide   7.211e-02  1.396e-02   5.164 2.41e-07 ***
## total_sulfur_dioxide -5.504e-02  5.522e-03  -9.967  < 2e-16 ***
## density               2.436e+03  2.320e+02  10.498  < 2e-16 ***
## pH                   -4.644e+00  1.589e+00  -2.922 0.003475 **
## sulphates             1.592e+00  1.382e+00   1.152 0.249313
## alcohol               2.782e+00  3.569e-01   7.797 6.36e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5843.79  on 5195  degrees of freedom
## Residual deviance:  331.46  on 5184  degrees of freedom
## AIC: 355.46
##
## Number of Fisher Scoring iterations: 9
```

At first sight, each of the coeffiecient has a marginal test which attempts the null hypothesis H_0: $\beta_i = 0$, after adjusting the coefficients within the model. That means, it is checked the net effect of each variable and whether should be in the model or not. All the p-values are small enough to reject H_0 (considering $\alpha = 0.05$) except for `citric_acid` (0.12) and `sulphates` (0.249). Let us discard these two variables in the further analysis:

```r
# Logistic regression model with all the variables.
log.F=glm(type~fixed_acidity+volatile_acidity+residual_sugar+chlorides
             +free_sulfur_dioxide+total_sulfur_dioxide+density
             +pH+alcohol, data=train, family=binomial)
summary(log.F)
```

```
##
## Call:
```

```
## glm(formula = type ~ fixed_acidity + volatile_acidity + residual_sugar +
##     chlorides + free_sulfur_dioxide + total_sulfur_dioxide +
##     density + pH + alcohol, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6691  -0.0555  -0.0226  -0.0004   5.4365
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -2.529e+03  2.094e+02 -12.080  < 2e-16 ***
## fixed_acidity        -1.102e+00  2.389e-01  -4.614 3.95e-06 ***
## volatile_acidity      5.677e+00  9.761e-01   5.816 6.02e-09 ***
## residual_sugar       -1.042e+00  1.003e-01 -10.385  < 2e-16 ***
## chlorides             1.894e+01  4.094e+00   4.627 3.72e-06 ***
## free_sulfur_dioxide   7.492e-02  1.379e-02   5.435 5.49e-08 ***
## total_sulfur_dioxide -5.620e-02  5.409e-03 -10.390  < 2e-16 ***
## density               2.538e+03  2.127e+02  11.930  < 2e-16 ***
## pH                   -4.596e+00  1.524e+00  -3.015  0.00257 **
## alcohol               2.922e+00  3.201e-01   9.129  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5843.79  on 5195  degrees of freedom
## Residual deviance:  335.69  on 5186  degrees of freedom
## AIC: 355.69
##
## Number of Fisher Scoring iterations: 9
```

All the marginal tests are now significantly small and the overall fit of the model is high enough ($p = 1$), so we do not have evidence to reject the model in favour of the simple constant model.

```
pchisq(log.F$deviance, log.F$df.residual, lower=F)
```

```
## [1] 1
```

Validation

We will define the following function to validate the correctness of the model. It basically counts the number of correctly classified observations, and it is divided by the total number of examples.

```
accuracy <- function(model, train, test){
  # Train error
  train_prob=model$fitted
  train_prob=ifelse(train_prob>0.5,1,0)
  d_train = table(train_prob, train$type)

  # Test error
  test_prob = predict(model, newdata = test, type = "response")
  test_prob = ifelse(test_prob>0.5,1,0)
  d_test = table(test_prob, test$type)

  train_accuracy = sum(diag(d_train))/sum(d_train)
```

```
  test_accuracy = sum(diag(d_test))/sum(d_test)

  return(list("train" = train_accuracy, "test" = test_accuracy))
}
```

```
log.F.error = accuracy(log.F, train, test)
log.F.error$train
```

```
## [1] 0.9948037
```

```
log.F.error$test
```

```
## [1] 0.9961538
```

Hence, with the full model we obtain an accuracy of 99% in both train and test dataset. This can be explained by the fact that the `type` of a wine is clearly defined by a combination of its chemical properties, as expected.

Simpler Model

Though we obtain a satisfactory accuracy using almost all the variables of the dataset, we would like to find out a simpler model, where just a few attributes were used. This would lead to a more understable model, easy to interpret and efficient. For instance, we could agree that an optimal model is the one which provides, at least, a 95% of correct classification.

Let us start with the simplest model: the constant model. Since we know that the classes are a bit unbalanced, let us start with the model which sets all the labels to 1.

```
log.B=glm(type~1, data=train, family=binomial)
accuracy(log.B, train, test)
```

```
## $train
## [1] 0.75
##
## $test
## [1] 0.7707692
```

A bit more than 75% of accuracy just by guessing that all the wines will be red. However, we are not using the chemical information. Let us now include one of the variables to the logistic model. Which one? The one which decreases the most the AIC. The AIC is a measure of the quality of different models, relative to each of the other models. Ideal for model selection.

The function `step` does this task for us: it chooses a model by AIC in a stepwise algorithm. We would use it in the forward direction: tt starts by the simplest constant model, and it tries to achieve the best model up to the full model, previously defined. Since we will go step by step, we will set the number of `steps` manually, to see what happens in each level.

```
#Stepwise algorithm
step(log.B, scope=list(upper=log.F), direction="forward", step=1)
```

```
## Start:  AIC=5845.79
## type ~ 1
##
##                        Df Deviance    AIC
## + total_sulfur_dioxide  1   2302.6 2306.6
## + volatile_acidity      1   3522.8 3526.8
## + chlorides             1   3829.2 3833.2
## + free_sulfur_dioxide   1   4178.3 4182.3
## + fixed_acidity         1   4594.0 4598.0
```

```
## + residual_sugar        1   4925.6 4929.6
## + density               1   4929.1 4933.1
## + pH                    1   5278.9 5282.9
## + alcohol               1   5836.3 5840.3
## <none>                      5843.8 5845.8
##
## Step:  AIC=2306.6
## type ~ total_sulfur_dioxide

##
## Call:  glm(formula = type ~ total_sulfur_dioxide, family = binomial,
##     data = train)
##
## Coefficients:
##          (Intercept)  total_sulfur_dioxide
##              4.44197              -0.06273
##
## Degrees of Freedom: 5195 Total (i.e. Null);  5194 Residual
## Null Deviance:       5844
## Residual Deviance: 2303  AIC: 2307
```

Looking at the output, we observe that the best attribute to build a logistic regression with just a single variable is the `total_sulfur_dioxide`. The accuracy of the logistic regression variable for both train and test datasets is 92%! So finally, `type` is almost a matter of sulfur in the liquid. This is, nevertheless, not a surprise, since the most correlated variable with respect to the `type` is also this one:

```
log.1=glm(type~total_sulfur_dioxide, data=train, family=binomial)
accuracy(log.1, train, test)
```

```
## $train
## [1] 0.9247498
##
## $test
## [1] 0.9292308
```

```
cor(df, df$type)
```

```
##                             [,1]
## fixed_acidity         0.48724238
## volatile_acidity      0.65270551
## citic_acid           -0.18644067
## residual_sugar       -0.34885917
## chlorides             0.51243909
## free_sulfur_dioxide  -0.47258513
## total_sulfur_dioxide -0.70037221
## density               0.39067606
## pH                    0.32934670
## sulphates             0.48776481
## alcohol              -0.03254018
## quality              -0.11889034
## type                  1.00000000
```

Let us include one more variable and see what happens:

```
#Stepwise algorithm
step(log.1, scope=list(upper=log.F), direction="forward", step=1)
```

```
## Start:  AIC=2306.6
```

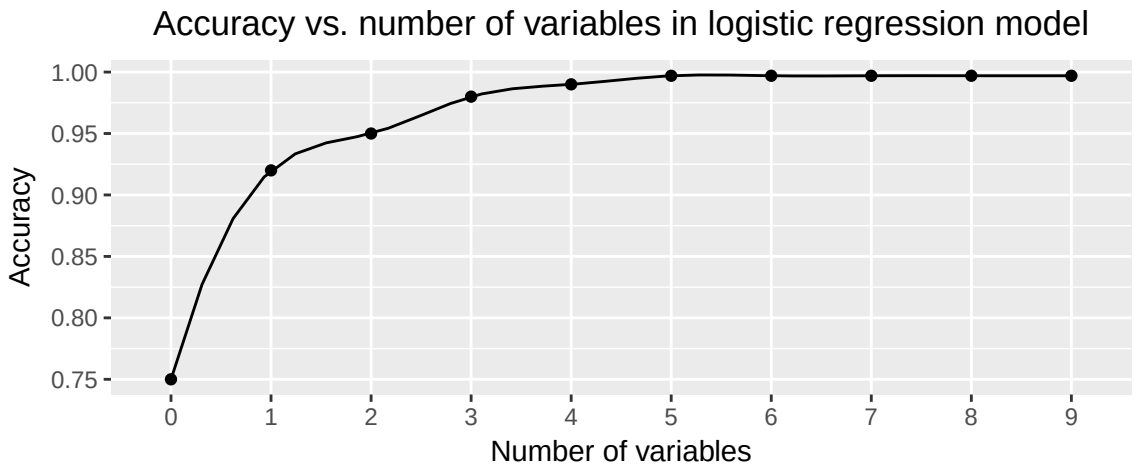
```
## type ~ total_sulfur_dioxide
##
##                       Df Deviance    AIC
## + density              1   1300.4 1306.4
## + volatile_acidity     1   1405.4 1411.4
## + chlorides            1   1551.1 1557.1
## + fixed_acidity        1   1942.0 1948.0
## + alcohol              1   2072.2 2078.2
## + pH                   1   2142.6 2148.6
## + residual_sugar       1   2279.6 2285.6
## + free_sulfur_dioxide  1   2282.7 2288.7
## <none>                     2302.6 2306.6
##
## Step:  AIC=1306.38
## type ~ total_sulfur_dioxide + density
##
## Call:  glm(formula = type ~ total_sulfur_dioxide + density, family = binomial,
##     data = train)
##
## Coefficients:
##          (Intercept)  total_sulfur_dioxide               density
##           -782.71663              -0.07262            792.08044
##
## Degrees of Freedom: 5195 Total (i.e. Null);  5193 Residual
## Null Deviance:       5844
## Residual Deviance: 1300  AIC: 1306
```

The AIC decays the most with the inclution of `density`, reching an accuracy of 95%. Using just two variables of the dataset, we are just wrong in 58 classifications (21 should have been red, and 37 white) up to a total of 1300 observations.

```
log.2=glm(type~total_sulfur_dioxide+density, data=train, family=binomial)
accuracy(log.2, train, test)
```

```
## $train
## [1] 0.9578522
##
## $test
## [1] 0.9553846
```

We stop here, since the addition of more variables does not significantly increase the accuracy of the model. The following plot summarises the accuracy of the logistic model with respect the number of variables included in the model, following the stepwise algorithm.


Accuracy vs. number of variables in logistic regression model
Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0
1
2
3
4
5
6
7
8
9
Number of variables